SteelEye FastAPI

Here's a breakdown of the solution and the reasoning behind the approach.

1. Importing Dependencies: The necessary libraries and modules are imported.
2. FastAPI App and Database: An instance of the FastAPI class is created, and an empty list called trades_db is used to store trade data.
3. TradeDetails and Trade Models: Two models, TradeDetails and Trade, are defined using Pydantic. These models specify the structure and validation rules for trade details and trades.
4. /trades Endpoint: The /trades route is defined using the @app.get decorator. It accepts query parameters for filtering trades based on different criteria. The trades are filtered based on the provided parameters and returned as a response.
5. /trades/{trade_id} Endpoint: The /trades/{trade_id} route is defined to retrieve a specific trade by its trade_id. If a trade with the given trade_id is found, it is returned. Otherwise, an error message is displayed.
6. Running the FastAPI Server: The FastAPI application is run using the uvicorn server, which starts on localhost at port 8000.

The reasoning behind the approach includes:

- Dependency Installation: The code includes an installation command for the required dependencies to make it easy for users to install them in a Jupyter Notebook environment.
- Separation of Components: The code is divided into separate cells to logically organize different parts of the application, making it easier to read and maintain.
- Models for Structured Data: Pydantic models are used to define the structure and validation rules for trade details and trades. These models ensure data consistency and provide automatic serialization and deserialization of JSON data.
- Endpoint Definition: The code uses decorators to define the routes and their associated functions. This approach allows the server to handle GET requests to specific routes and execute the corresponding functions.
- Filtering and Error Handling: The code includes logic to filter trades based on query parameters and handle cases where a trade is not found. It raises an appropriate error message using the HTTPException class.
- Running the Server: The uvicorn server is used to run the FastAPI application, enabling access to the defined endpoints.